Partitioning a Weighted Partial Order

Linda S. Moonen and Frits C.R. Spieksma
Department of Applied Economics, Katholieke Universiteit Leuven, Belgium.
{linda.moonen | frits.spieksma}@econ.kuleuven.be

Abstract

The problem of partitioning a partially ordered set into a minimum number of chains is a well-known problem. In this paper we study a generalization of this problem, where we not only assume that the chains have bounded size, but also that a weight w_i is given for each element i in the partial order such that $w_i \leq w_j$ if $i \prec j$. The problem is then to partition the partial order into a minimum-weight set of chains of bounded size, where the weight of a chain equals the weight of the heaviest element in the chain. We prove that this problem is $\mathcal{APX}$-hard, and we propose and analyze lower bounds for this problem. Based on these lower bounds, we exhibit a 2-approximation algorithm, and show that it is tight. We report computational results for a number of real-world and randomly generated problem instances.

1. Introduction

Consider a partially ordered set $(X, \prec)$. We say that two elements $i, j \in X$ are *comparable* if either $i \prec j$ or $j \prec i$. A *chain* C is defined as a subset of X such that all elements $i, j \in C$ are pairwise comparable. An *antichain* A is a subset of X such that no two elements $i, j \in A$ are comparable. The *size* of a chain (or antichain) is equal to the number of elements contained in it (Trotter (1992)).

Now, the problem of partitioning a partially ordered set $(X, \prec)$ into a minimal number of chains such that each element of X belongs to at least one chain, is a well-known, fundamental problem in operations research. This problem is solvable in polynomial time, and the size of a maximum antichain is equal to the minimum number of chains needed to cover all elements of X (Dilworth (1950)).

Shum and Trotter (1996) generalize this problem by assuming that an integer B is given that bounds the size of a chain. Thus, in this setting no more than B elements can be in a chain. They show that the corresponding decision problem is $\mathcal{NP}$-complete, even for a fixed $B = 3$.

In this work we further generalize this problem by assuming that a weight w_i for each $i \in X$ is given such that $w_i \leq w_j$ if $i \prec j$. Moreover, we define the weight of a chain C as $\max_{i \in C} w_i$, thus, the weight of a chain is equal to the weight of the heaviest element in the chain. Further, we denote a chain containing at most B elements as a B-chain. The problem is now to partition X into a minimum-weight set of B-chains. We refer to this problem as Minimum Weight Partition into B-chains, or MWPB. Observe that when $w_i = 1$ for all $i \in X$ the problem dealt with by Shum and Trotter (1996) arises.

1.1 Applications

Applications of MWPB can be found in the field of mutual exclusion scheduling (Baker and Coffman (1996), Jansen (2003)), also known as batch scheduling with job compatibilities (Boudhar (2003), Finke et al. (2004)). In such a problem jobs are given, each with a given processing time p_i, and for each pair of jobs it is known whether they can be processed on a same machine. A machine can process at most B jobs simultaneously, and the time a

machine needs to process its jobs equals the maximum processing time of the jobs assigned to that machine. The problem is then to assign the jobs to the machines, respecting the compatibilities, while minimizing the largest completion time. To represent the job compatibilities, often a graph is used; different types of graphs lead to different complexity results. In our setting, the graph corresponding to the job compatibilities is a comparability graph (see e.g. Golumbic (1980)).

Another application of MWPB is described in Moonen and Spieksma (2005). Given are a number of rectangular shaped boxes, each with a given length ℓ_i and width w_i. These boxes need to be loaded on pallets, and each pallet can hold at most B boxes. Moreover, for any pair of consecutive boxes on a pallet it must hold that $\ell_i \leq \ell_j$ and $w_i \leq w_j$. The goal is to load all boxes onto pallets so as to minimize total area, where the area of a pallet is determined by the area of its largest item. Observe that this corresponds to a special case of MWPB, where the special case arises since the partial order induced by the lengths and widths (i.e., $i \prec j$ if and only if $\ell_i \leq \ell_j \wedge w_i \leq w_j$) can be embedded in two dimensions; in other words, the dimension of the partial order equals 2 (see Ore(1962)).

1.2 Our results

In this paper we show the following:

- Strengthening a result from Shum and Trotter (1996), we show that MWPB is $\mathcal{APX}$-hard, rendering the existence of a PTAS unlikely.
- We propose two lower bounds, each of which can be arbitrarily bad when compared to the value of the optimum. The maximum of these lower bounds, however, is shown never to be less than half the optimum value.
- We describe a simple algorithm that yields a solution with a value guaranteed not to exceed twice the optimum value. The analysis is shown to be tight.
- We consider an extension of the special case of MWPB where the dimension is 2, to a setting where rotation of items is allowed.

The outline of the paper is as follows. Section 2 deals with the complexity of MWPB. In Section 3 we propose a number of lower bounds on the value of the optimum, and in Section

4 we present a 2-approximation algorithm for solving MWPB. We tested the algorithm on a number of real-world problem instances, as well as on randomly generated instances, and the results from these experiments are described in Section 5. In Section 6 we discuss an interesting variant of MWPB, where the orientation of an element is taken into account. Finally, in Section 7, we conclude.

2. Complexity of MWPB

The decision problem corresponding to MWPB can be formulated as follows:

Given an integer B, a partial order $(X, \prec)$, weights w_i with $w_i \leq w_j$ if $i \prec j$ $(\forall i, j \in X)$, and an integer K, does there exist a partition of X into B-chains such that the sum of the weights of the B-chains does not exceed K?

As stated in Section 1, Shum and Trotter (1996) prove that this decision problem is $\mathcal{NP}$-complete, even if $w_i = 1$, for all $i \in X$. We can strengthen their result:

Theorem 1. *MWPB is $\mathcal{APX}$-hard, even if*

- *$B = 3$, and*
- *$w_i = 1$ $\forall i$, and*
- *each element occurs in no more than 3 chains.*

Proof: We use a reduction from the Maximum 3-bounded 3-dimensional matching problem (3DM-3), and follow the reduction from Shum and Trotter (1996). Furthermore, we apply arguments used in Chekuri and Khanna (2005).

Problem 3DM-3 is defined as follows: given are three sets X, Y, Z with $|X| = |Y| = |Z| = n$, and a set of triples $T \subseteq X \times Y \times Z$ with $|T| = m$ (we assume that $m = O(n)$). The goal is to find a matching of largest cardinality. Kann (1991) showed that this problem is $\mathcal{APX}$-hard, so it is $\mathcal{NP}$-hard to decide whether there exists a matching of size n, or whether every matching has size at most $(1 - \delta)n$ for some fixed $\delta > 0$.

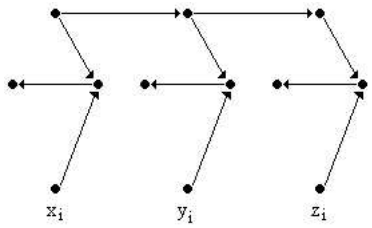

Figure 1: Subgraph for triple $t_i = \{x_i, y_i, z_i\}$, see Shum and Trotter (1996).

Now, consider an instance $\mathcal{I}$ of problem 3DM-3, and build the corresponding instance $\mathcal{I}'$ of problem MWPB with $B = 3$ as described in Shum and Trotter (1996) (See also Garey and Johnson (1979), page 69): to each triple $t_i \in T$, $t_i = \{x_i, y_i, z_i\}$, we associate a subgraph (called a *configuration*) as shown in Figure 1, where an arc from i to j implies that $i \prec j$. Observe that

i) each chain in the instance $\mathcal{I}'$ must be contained within a single configuration, and

ii) only if x_i and y_i and z_i are covered by other chains, it is possible to use 3 chains (recall that $B = 3$) for the points in the configuration; otherwise at least 4 chains are needed.

If instance $\mathcal{I}$ has a matching of size n, then there exists a solution to instance $\mathcal{I}'$ with $4n + 3(m - n) = n + 3m$ chains: we need 4 chains to cover each element in a configuration that corresponds to a chosen triple in the matching. For the elements in the remaining configurations, we need 3 chains to cover them, since all x_i, y_j, and z_k are covered by other chains.

Now let us consider the case that every matching has size at most $(1-\delta)n$ for some fixed $\delta > 0$. First of all, we need at most $4(1-\delta)n$ chains to cover the elements contained in configurations corresponding to triples in the matching. Then we have covered at most $3(1-\delta)n = 3n-3\delta n$ x, y, z-elements. So there are at least $3\delta n$ x, y, z-elements remaining. Observe that there can be no configuration that contains more than 2 of these $3\delta n$ elements, since otherwise a better solution (i.e., a matching exceeding size $(1 - \delta)n$) exists. That means that the minimum number of configurations needed to cover these elements is $\frac{3\delta n}{2}$, and we need at least $4(\frac{3\delta n}{2})$ chains to cover all elements in these configurations. Finally, for the remaining elements we need at least 3 times the number of remaining configurations $= 3(m-(1-\delta)n-\frac{3\delta n}{2})$ chains. So in total, we need at least $4(1 - \delta)n + 4(\frac{3\delta n}{2}) + 3(m - (1 - \delta)n - \frac{3\delta n}{2}) = n + 3m + \frac{1}{2}\delta n$ chains. Since $m = O(n)$, the $\mathcal{APX}$-hardness follows. □

3. Lower bounds for MWPB

Consider an instance of MWPB, containing an integer B and n elements, each with a weight w_i, $1 \leq i \leq n$. We assume that the elements are ordered such that $w_1 \geq w_2 \geq \cdots \geq w_n$. Let OPT denote the value of an optimal solution to the instance. We define three lower bounds lb_i, $i = 1, 2, 3$, as follows:

1. $lb_1 = w_1 + w_{B+1} + \ldots + w_{(\lceil \frac{n}{B} \rceil - 1)B+1}$. Since the size of a chain cannot exceed B, lb_1 is obviously a lower bound for OPT.

2. When we omit the size constraint (i.e., if there is no restriction on the size of a chain), a relaxation of MWPB appears. Solving this relaxation gives a minimum-weight set of chains with value MWC. We set $lb_2 = MWC$.

3. $lb_3 = \max(lb_1, lb_2)$.

Theorem 2. *We can calculate the value of lb_2 by solving a min-cost flow problem.*

Proof: In order to compute the value of lb_2, we create a directed graph $D = (V, A)$. V contains $2n + 2$ nodes: 2 nodes i' and i'' for every $i \in X$, a source s and a sink t. We draw an arc from s to each node i', with cost 0. Then we add an arc from each node i'' to t with cost w_i. Next, we add an arc from a node i' to its copy i'' with cost 0, and we add arcs from nodes i'' to j' if $i \prec j$, also with cost 0. Finally we add an arc from s to t with cost 0. All nodes have supply zero, except for s which has supply n, and t, which has supply $-n$ (a demand of n). All arc capacities are equal to 1, and for the arcs from a node i' to its copy i'' we have a lower bound on the flow of 1. Now, a min-cost flow in D can be easily translated to a solution to MWPB without the size constraint and vice versa. □

Notice that this algorithm solves a weighted generalization of the classical result of Dilworth (1950).

Example – Min-Cost Flow

Suppose we are given a partial order S containing four points, $S = \{a, b, c, d\}$, and suppose $a \prec b$, $a \prec d$ and $c \prec d$. The min-cost flow network corresponding to this example is shown

in Figure 2. (The lower and upper bounds on the arcs, the arc costs and the demands are omitted from this figure.)

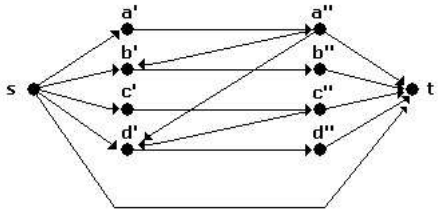

Figure 2: Min-Cost Flow Network.

If we solve this min-cost flow problem, we find a solution containing a number of paths that have positive flow value. In the solution to our original problem, we put two points in the same chain if, in the solution to the min-cost flow problem, these points appear on the same path with positive flow value. This corresponds to a minimum-weight set of chains, keeping in mind that we have disregarded the size requirement on the chains. □

We now show that lb_1 and lb_2 can be arbitrarily bad, even in the unweighted case. Consider lb_1, and suppose we are given a problem instance with $B = n$, such that no two elements are comparable, and suppose that each element has weight 1. One easily verifies that OPT equals n, while lb_1 equals 1.

Next, consider lb_2, and suppose we have a problem instance with $B = 1$, such that all elements are comparable, and that each element has weight 1. Again, OPT equals n, while lb_2 gives a value of 1. So, we cannot give a constant performance guarantee for either of these lower bounds. However, no instance exists where both lower bounds are arbitrarily bad. Indeed, let us now consider the maximum of these two lower bounds, lb_3.

Claim 1. *For each instance of MWPB: $lb_3 \geq \frac{1}{2}OPT$. Moreover, this bound is tight.*

We postpone the proof of this inequality to Theorem 3; we first give an instance for which this bound is tight.
Consider the problem instance given in Figure 3, with $n = B^2$. So, we have B^2 elements, all with weight 1, such that $n - \frac{n}{B} + 1$ of these elements are pairwise comparable, and the remaining $\frac{n}{B} - 1$ elements are pairwise incomparable. Observe that, for such instances, $lb_3 = max(lb_1, lb_2) = B$, while $OPT = (\frac{n}{B} - 1) + \lceil (\frac{n - \frac{n}{B} + 1}{B}) \rceil = (B-1) + \lceil \frac{B^2 - B + 1}{B} \rceil = (2B-1)$.

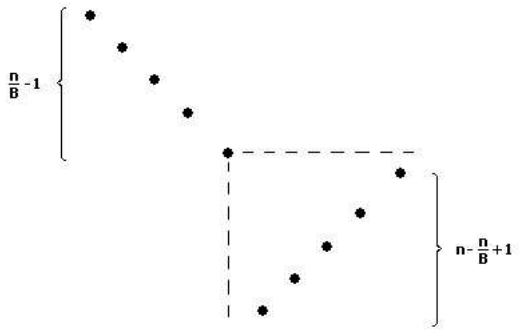

Figure 3: A Tight Example.

4. A 2-approximation algorithm for MWPB

In this section we propose a 2-approximation algorithm for MWPB, and show that it is tight.

Consider the following heuristic H:

Step 1. Omit the size constraint, and find a minimum-weight set of chains as described in Theorem 2.

Step 2. For each chain consisting of say K elements $i_1, \ldots, i_K$, with $i_1 \succ i_2 \succ \ldots \succ i_K$, partition it into $\lceil \frac{K}{B} \rceil$ B-chains such that elements $i_{(j-1)B+1}, i_{(j-1)B+2}, \ldots, i_{\min(jB,K)}$ form B-chain j, $j = 1, \ldots, \lceil \frac{K}{B} \rceil$.

Theorem 3. *H is a 2-approximation algorithm for problem MWPB.*

Proof: We assume that the elements are ordered such that $w_1 \geq w_2 \geq \cdots \geq w_n$. Suppose we find a solution using heuristic H with value v_H, where in the first step we find a decomposition into p chains, $C_1, ..., C_p$. In the second step we partition each of these p chains into a number of B-chains. The maximal elements of the B-chains that contain the maximal elements of C_ℓ are referred to as i_ℓ, $1 \leq \ell \leq p$. All other maximal elements of B-chains are referred to as j_ℓ, $1 \leq \ell \leq k$. Assume, without loss of generality, that $j_1 \leq j_2 \leq \ldots \leq j_k$. Notice that we can associate to each item j_ℓ a set of B items that belong to the same chain found in Step 1 as j_ℓ, and are the smallest B items that dominate j_ℓ. Let us refer to this set of items as $S(j_\ell)$, $1 \leq \ell \leq k$.

Claim 2. $j_\ell \geq \ell B$

Argument: Consider the sets $S(j_\ell)$, $\ell = 1, \ldots, k$. Since these sets are pairwise disjoint, the number of items that must precede j_ℓ, $1 \leq \ell \leq k$, equals at least ℓB. □

The inequality from Claim 2 implies $\sum_{\ell=1}^{k} w_{j_\ell} \leq \sum_{\ell=1}^{k} w_{\ell B} \leq lb_1$. And, obviously, $\sum_{\ell=1}^{p} w_{i_\ell} = lb_2$. Thus, we have that $v_H = \sum_{\ell=1}^{p} w_{i_\ell} + \sum_{\ell=1}^{k} w_{j_\ell} \leq lb_1 + lb_2 \leq 2OPT$. Also, notice that $lb_3 + lb_3 \geq lb_1 + lb_2 \geq v_H \geq OPT$, implying Theorem 3. □

It is not clear yet whether the bound derived is tight. Indeed, the example for which lb_3 is shown to be worst possible is solved to optimality by H. We know that a solution found by heuristic H can be no worse than twice the optimal value. Can we find problem instances for which this gap is tight?

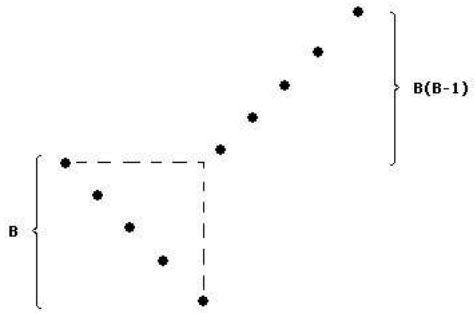

Figure 4: Worst-case instance with respect to H.

For the problem instances shown in Figure 4, we have $n = B^2$ elements, all with weight 1, and there are B elements that are pairwise incomparable, and the remaining $B(B-1)$ elements are pairwise comparable. For such an instance, we have $OPT = lb_1 = lb_2 = lb_3 = B$, while the heuristic H gives a solution with value $(B-1) + \lceil \frac{B^2-B+1}{B} \rceil = (2B-1)$, so these are asymptotic worst-case instances with respect to H.

5. Computational results

We implemented the 2-approximation algorithm in C++, using the CPLEX network solver to solve the min-cost flow problems, and we tested it on a number of real-world and randomly generated problem instances. We use 3 different data sets for the experiments. The first

data set contains 50 real-world instances provided to us by Bruynzeel Storage Systems, the second data set contains 50 randomly generated instances, and the third data set contains 50 randomly generated instances that have small clique-width (see Moonen and Spieksma (2005)). The problem instances for all three data sets contain between 20 and 200 elements. We solve each problem instance for 5 different values of B, so we have 250 experiments for each data set. (In the real-world setting of Bruynzeel Storage System, B equals 12.) Since the computation times were 0.00 seconds, for all instances, we omit them from the tables with results.

Table 1: Results for lower bounds

B	*Data set 1*		*Data set 2*		*Data set 3*	
	$lb_3 = lb_1$	$lb_3 = lb_2$	$lb_3 = lb_1$	$lb_3 = lb_2$	$lb_3 = lb_1$	$lb_3 = lb_2$
3	100%	0%	96%	4%	96%	4%
6	100%	0%	74%	26%	74%	26%
9	98%	2%	50%	50%	28%	72%
12	94%	6%	42%	58%	14%	86%
15	94%	6%	26%	74%	8%	92%

In Table 1 we compare the lower bounds. As lb_3 is defined as the maximum of lb_1 and lb_2, we want to know how many times lb_3 equals lb_1, and how many times it equals lb_2. So in Table 1 we give, for each of the three data sets, the percentage of the number of times that lb_3 equals lb_1 and the number of times that lb_3 equals lb_2.

Table 2: Results for data set 1: real-world instances

B	lb_1	lb_2	lb_3	v_H	Δ_3 (%)	
					avg_{Δ_3}	max_{Δ_3}
3	1085.06	96.02	1085.06	1096.12	1.41	5.21
6	556.12	96.02	556.12	574.74	3.85	14.81
9	379.96	96.02	380.22	396.66	5.51	21.60
12	293.76	96.02	294.82	316.08	7.04	24.27
15	240.40	96.02	242.18	259.64	7.78	20.10

Tables 2, 3, and 4 show a comparison between the values of the three lower bounds and the value of the 2-approximation algorithm. In the first column we give the value of B, and in the next four columns we show the average values of the three lower bounds (lb_1, lb_2, and lb_3) and the 2-approximation algorithm (v_H). Of course, as can be seen in the third column, the

Table 3: Results for data set 2: random instances

B	lb_1	lb_2	lb_3	v_H	Δ_3 (%)	
					avg_{Δ_3}	max_{Δ_3}
3	6518.64	1508.44	6589.06	6935.18	9.72	24.09
6	3317.18	1508.44	3338.78	3877.36	15.66	26.17
9	2259.08	1508.44	2314.28	2867.24	15.03	27.10
12	1726.68	1508.44	1818.90	2348.64	14.21	30.51
15	1411.90	1508.44	1585.92	2066.94	13.05	31.92

Table 4: Results for data set 3: instances with small clique-width

B	lb_1	lb_2	lb_3	v_H	Δ_3 (%)	
					avg_{Δ_3}	max_{Δ_3}
3	3215.70	1357.98	3219.66	3607.64	11.13	22.76
6	1686.28	1357.98	1765.54	2200.00	18.57	27.85
9	1177.96	1357.98	1451.42	1762.92	15.96	30.33
12	926.42	1357.98	1389.50	1574.78	11.15	27.43
15	775.38	1357.98	1364.50	1470.34	7.44	28.52

value of B does not influence lb_2. Finally, the column labelled Δ_3 shows the average (avg_{Δ_3}) and the maximum (max_{Δ_3}) difference between the values of v_H and lb_3. These differences are all given in percentages (i.e., $\frac{v_H - lb_3}{v_H} \cdot 100\%$).

From these results we see that the performance of the lower bounds is very different for the different data sets. For the first data set, that contains the real-world problem instances, lb_1 is clearly better than lb_2: in 97.20% of all experiments, the value of lb_1 is larger than the value of lb_2. If we look at the second data set, we see that lb_1 still performs better compared to lb_2, but the percentage of experiments for which lb_1 is larger than lb_2 in only 57.60% for data set 2. However, for data set 3 we see that lb_2 performs slightly better than lb_1: in 56.00% of all experiments the value of lb_2 is larger than the value of lb_1.

Next we compare the values of lb_3 with the values of the approximation algorithm. The difference between the value of the approximation algorithm and the value of lb_3 could get as large as 100%, however, the maximum difference among all experiments from the three data sets is equal to 24.27% for data set 1, 31.92% for data set 2, and 30.33% for data set 3. The average difference for the three data sets equal 5.12% for data set 1, 13.53% for data set 2, and 12.85% for data set 3.

6. Rotation Problem

In some applications, for example in the field of pallet loading problems as discussed in Moonen and Spieksma (2005), the items are allowed to be *rotated* (i.e., the length and the width of an item are swapped) if that allows us to find a better solution. It is not difficult to exhibit instances where allowing rotation improves the solution value. So a relevant problem is, when rotation is allowed, to choose the best orientation for each item. Of course this question is motivated by a two-dimensional representation of the items. However, each partial order can be embedded in d-dimensional space for some d, and, therefore the rotation variant of MWPB is interesting in higher dimensions as well. In this variant the objective is still to partition X into a minimum-weight set of B-chains. However, now we are allowed to choose an orientation for each element of X. We refer to this problem as MWPB-R.

Theorem 4. *There exists an optimal solution to MWPB-R such that, for all points p that are contained in this optimal solution, it holds that*

$$x_p^1 \geq x_p^2 \geq \ldots \geq x_p^d \tag{1}$$

where d is the number of dimensions, and x_p^i is the i-th coordinate of point p.

Informally said, Theorem 4 states that there is no loss in rotating each item such that for each item the i-th largest coordinate becomes its size in the i-th dimension. Before we give a proof of this theorem, we give an example for the 2-dimensional case.

Example – The 2-dimensional case

Given are n pairs of points $\{(x_i^1, x_i^2), (x_i^2, x_i^1)\}$, $i = 1, \ldots, n$ in the 2-dimensional plane. Exactly one point from each of these n pairs must be present in a solution. Suppose we have an optimal solution that contains the chain $C = \{P, Q, R\}$ (see Figure 5). Theorem 4 states that there exists an optimal solution such that for all the selected points in this solution we have that $x_i^1 \geq x_i^2$. That means that we can exchange all the points that lie above the line $x^1 = x^2$ with their corresponding copies that lie below the line $x^1 = x^2$. This corresponds to the chain $C' = \{P, Q', R\}$. To see why this is true, consider the following:

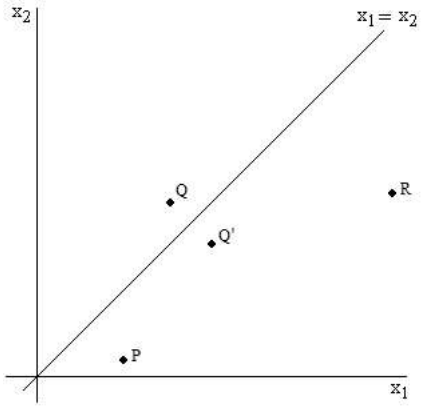

Figure 5: Example for the 2-dimensional case.

Since the chain containing points P, Q, and R is in the original solution, we know that $P \prec Q \prec R$, and so we have:

$$x_P^1 \leq x_Q^1 \leq x_R^1$$
$$x_P^2 \leq x_Q^2 \leq x_R^2$$

We also know that points P and R lie below the line $x^1 = x^2$, and point Q lies above this line, so we also have:

$$x_P^1 \geq x_P^2$$
$$x_Q^1 \leq x_Q^2$$
$$x_R^1 \geq x_R^2$$

Now, in order for C' to be feasible, we must have that $P \prec Q' \prec R$, so we must show that

$$x_P^1 \leq x_Q^2 \leq x_R^1 \tag{2}$$
$$x_P^2 \leq x_Q^1 \leq x_R^2 \tag{3}$$

Equation (2) is true since $x_P^1 \leq x_Q^1 \leq x_Q^2 \leq x_R^2 \leq x_R^1$. Equation (3) is true since $x_P^2 \leq x_P^1 \leq x_Q^1 \leq x_Q^2 \leq x_R^2$. □

Now we continue with the proof of Theorem 4.

Proof: In order to prove Theorem 4, we must show that if we have a chain C containing points that do not satisfy condition (1), then the copies of these points that do satisfy condition (1) form a chain. So, given two arbitrary points U and V such that $U \prec V$, we must show that, for their copies satisfying condition (1), called U' and V', it holds that $U' \prec V'$. So we have to prove that if $U \prec V$, then $U' \prec V'$. That means that we have to show, for each $i = 1, \ldots, m$, that $x^i_{U'} \leq x^i_{V'}$.

Take the smallest i for which this does not hold, so we have $x^i_{V'} < x^i_{U'}$. This means that the i^{th}-largest coordinate of point V' is smaller than the i^{th}-largest coordinate of point U', which contradicts $U \prec V$. □

Observe that in the argument above B plays no role. Hence, using Theorem 2, we can solve instances of MWPB-R with $B \geq n$ to optimality in polynomial time.

Notice that rotating the items as described by (1) increases the number of pairs of items that are comparable. It would be interesting to see how this would influence the performance of lb_2 and the heuristic H on the problem instances of Section 5.

7. Conclusions

In this paper we discuss the problem of partitioning a weighted partially ordered set into chains of bounded size. We proposed three lower bounds for this problem, and presented a 2-approximation algorithm for solving it. The approximation algorithm is tested on a number of real-world and randomly generated problem instances. From the results of the experiments we see that, although the value of heuristic H could be up to twice the value of lb_3, the largest difference between these values over all 750 experiments is 31.92%, and the average difference equals 10.50%. We conclude from these results that the approximation algorithm performs reasonably well.

References

Baker, B.S. and E.G. Coffman, Jr. (1996). Mutual exclusion scheduling. *Theoretical Computer Science* **162** 225-245.

Boudhar, M. (2003). Scheduling a batch processing machine with bipartite compatibility graphs. *Mathematical Methods of Operations Research* **57** 513-527.

Chekuri, C. and S. Khanna (2005). A PTAS for the multiple knapsack problem. To appear in *SIAM Journal on Computing*.

Dilworth, R.P. (1950). A decomposition theorem for partially ordered sets. *Annals of Mathematics* **51** 161-166.

Finke, G., V. Jost and M. Queyranne (2004). Batch processing with interval graph compatibilities between tasks. In *Proceedings of Discrete Optimization Methods in Production and Logistics 2004*.

Garey, M.R. and D.S. Johnson (1979). *Computers and intractability. A guide to the theory of NP-completeness.* W.H. Freeman and Company, New York.

Golumbic, M.C. (1980). *Algorithmic graph theory and perfect graphs.* Academic Press, New York.

Jansen, K. (2003). The mutual exclusion scheduling problem for permutation and comparability graphs. *Information and Computation* **180** 71-81.

Kann V. (1991). Maximum bounded 3-dimensional matching is MAX SNP-complete. *Information Processing Letters* **37(1)** 27-35.

Moonen, L.S. and F.C.R. Spieksma (2005). Exact algorithms for a loading problem with bounded clique width. To appear in *INFORMS Journal on Computing*.

Ore, O. (1962). *Theory of Graphs.* American Mathematical Society Colloquium Publications, Volume 38, Providence, RI.

Shum, H. and L.E. Trotter, Jr. (1996). Cardinality-restricted chains and antichains in partially ordered sets. *Discrete Applied Mathematics* **65** 421-439.

Trotter, W.T. (1992). *Combinatorics and partially ordered sets: dimension theory.* The John Hopkins University Press, Baltimore.